UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/658,593	09/08/2003	Arndt Rosenthal	09700.0075-00	1930

60668 7590 06/20/2008

SAP / FINNEGAN, HENDERSON LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
YIGDALL, MICHAEL J	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
06/20/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/658,593	ROSENTHAL ET AL.	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 March 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,5-9,11-14 and 16-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,5-9,11-14 and 16-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is responsive to Applicant's reply filed on March 14, 2008. Claims 1, 2, 5-9, 11-14 and 16-18 are pending.

Response to Amendment

2. The provisional obviousness-type double patenting rejection set forth in the last Office action is withdrawn in view of Applicant's amendments to the claims.

Response to Arguments

3. Applicant's arguments have been considered but are moot in view of the new ground(s) of rejection, as set forth below with reference to Quinn. Applicant's amendments to the claims necessitated the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 5-8, 11, 12, 14, 16 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 7,007,278 to Gungabeesoon (art of record, "Gungabeesoon") in view of U.S. Patent No. 6,449,617 to Quinn et al. (now made of record, "Quinn") and in view of U.S. Patent No. 7,003,482 to Margoscin et al. (art of record, "Margoscin").

With respect to claim 1 (currently amended), Gungabeesoon discloses a computer program product, tangibly embodied in a machine-readable storage device, the computer program product being operable to cause data processing apparatus to perform operations (see, for example, column 7, lines 2-19) comprising:

receiving run-time code for an application (see, for example, column 11, lines 2-7, which shows receiving run-time code for an application in the form of a JavaServer Page, and column 10, lines 42-49, which shows that the run-time code is for a legacy application), the run-time code being generated from a converted design-time representation of the application (see, for example, column 9, lines 14-17, which shows that the run-time code is generated from converted design-time representations of the application in the form of user interface pages).

Gungabeesoon does not expressly disclose the run-time code comprising a flag indicating that the run-time code was generated from the converted design-time representation.

Nonetheless, one of ordinary skill in the art could, with predictable results, implement the teachings of Gungabeesoon such that the run-time code comprises such a flag. For example, in an analogous art, Quinn teaches HTML code that includes a flag identifying the application that generated the code (see, for example, column 6, lines 21-62, and column 8, lines 44-63). The flag facilitates the appropriate handling of the code based on the application that generated the code (see, for example, column 5, line 66 to column 6, line 11).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the teachings of Gungabeesoon such that the run-time code comprises a flag indicating that the run-time code was generated from the converted design-time representation, as Quinn suggests.

Gungabeesoon further discloses that:

the converted design-time representation of the application is generated from an original design-time representation of the application developed for use in a first run-time environment for executing applications having been developed in a first design-time environment (see, for example, column 8, lines 44-54, which shows that the converted design-time representations are generated from original design-time representations of the application in the form of screen definitions, and column 7, lines 50-56, which shows a first run-time environment for executing applications developed in a first design-time environment), the converted design-time representation is stored as metadata in a metadata repository (see, for example, column 9, lines 41-48 and 54-60, which shows that the converted design-time representation is metadata for application data inserted at run-time, and column 11, lines 2-7, which further shows that the converted design-time representation is stored in a repository for retrieval at run-time), the converted design-time representation having been developed by a development tool based on a metamodel that defines metamodel objects (see, for example, column 8, line 67 to column 9, line 7, and column 9, line 54 to column 19, line 2, which shows that the converted design-time representation is developed based on a metamodel that defines metamodel objects such as HTML code, JavaScript code and JavaBean data objects), the first design-time environment using a first programming model comprising one or more first model elements including screens and processing logic for each screen (see, for example, column 8, lines 12-29, which shows that the first design-time environment uses a programming model comprising screens and processing logic), the original design-time representation including one or more application screens and original processing logic for each application screen (see, for example, column 8, lines 44-47,

which shows that the original design-time representation includes one or more application screens), the original processing logic including a call to a run-time module in the first run-time environment (see, for example, column 8, lines 29-32, which shows that the original processing logic includes calls to run-time modules in the first run-time environment); and

the converted design-time representation of the application is for use in a second run-time environment for executing applications having been developed in a second design-time environment (see, for example, column 9, lines 41-48, which shows a second run-time environment for executing applications developed in a second design-time environment), the second design-time environment using a second programming model comprising one or more second model elements including models, views, and controllers (see, for example, column 8, line 67 to column 9, line 7, which shows that the second design-time environment uses a programming model that comprises views in the form of HTML code and controllers in the form of JavaScript code, and column 9, line 54 to column 10, line 2, which shows that the programming model comprises models in the form of JavaBean data objects), the converted design-time representation including one or more application views based on the one or more application screens, and converted processing logic based on the original processing logic, the converted processing logic being capable of being executed in the second run-time environment (see, for example, column 10, lines 31-36, which shows that the converted design-time representation includes one or more application views and processing logic based on the original application screens and processing logic, and column 10, lines 42-49, which shows that the converted processing logic is executable in the second run-time environment).

Gungabeesoon further discloses:

executing the run-time code in the second run-time environment using an adapter operable to interface with the run-time module in the first run-time environment (see, for example, column 11, lines 8-22, which shows executing the run-time code in the second run-time environment and interfacing with the first run-time environment, and column 10, lines 3-17, which shows that the interface is an adapter in the form of a Publish-to-Web component), comprising using the adapter to perform a function not performed by the original processing logic (see, for example, column 9, lines 17-27, which shows that executing the run-time code comprises using the adapter to perform functions that the original processing logic does not perform).

Gungabeesoon does not expressly disclose using the adapter to perform a function comprising input validation.

However, Gungabeesoon does teach performing input validation (see, for example, column 8, line 67 to column 9, line 7). One of ordinary skill in the art could, with predictable results, implement the teachings of Gungabeesoon such that the adapter performs the input validation. For example, in an analogous art, Margoscin teaches a middleware program that functions as an adapter operable to interface with a user interface and a business transaction server (see, for example, the abstract). The middleware program (i.e., the adapter) performs support functions such as data and syntax validation (see, for example, column 3, lines 14-19). Specifically, Margoscin teaches that the middleware program (i.e., the adapter) performs a function comprising input validation (see, for example, column 6, lines 20-36).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the teachings of Gungabeesoon and Quinn such that executing

the run-time code comprises using the adapter to perform a function comprising input validation not performed by the original processing logic, as Margoscin suggests.

With respect to claim 5 (previously presented), the rejection of claim 1 is incorporated, and Gungabeesoon further discloses that the function comprises input formatting (see, for example, column 9, lines 28-31, which shows performing input formatting).

With respect to claim 6 (original), the rejection of claim 1 is incorporated, and Gungabeesoon further discloses that:

the original design-time representation of the application comprises original state control logic (see, for example, column 7, lines 60-66, which shows that the application comprises original state control logic); and

the converted design-time representation of the application comprises converted state control logic based on the original state control logic, the converted state control logic capable of being executed by the adapter (see, for example, column 8, lines 5-11, which shows that the converted design-time representation comprises converted state control logic, and column 11, lines 8-22, which shows that the adapter executes the converted state control logic).

With respect to claim 7 (original), the rejection of claim 1 is incorporated, and Gungabeesoon further discloses that:

the original design-time representation of the application comprises one or more controls from a first set of controls (see, for example, column 8, lines 54-57, which shows that the original design-time representation comprises one or more user interface elements or controls);

the converted design-time representation of the application comprises one or more controls from a second set of controls, each control in the converted design-time representation of the application corresponding to a control in the original design-time representation of the application (see, for example, column 9, lines 54-60, which shows that the converted design-time representation comprises one or more controls corresponding to the controls in the original design-time representation); and

executing the run-time code comprises rendering the controls in the converted design-time representation of the application (see, for example, column 8, lines 44-54, which shows rendering the controls).

With respect to claim 8 (currently amended), Gungabeesoon discloses a computer program product, tangibly embodied in a machine-readable storage device, the computer program product being operable to cause data processing apparatus to perform operations (see, for example, column 7, lines 2-19) comprising:

receiving run-time code for an application (see, for example, column 11, lines 49-55, which shows run-time code for an application comprising network application data and legacy application data, and see, for example, column 10, lines 37-42, and column 11, lines 2-7, which shows receiving such run-time code from a servlet instance);

determining whether the run-time code was generated from a native design-time representation of the application or from a converted design-time representation of the application (see, for example, column 10, lines 42-49, and column 11, lines 8-22, which shows the servlet instance responding in different ways depending on whether the run-time code comprises network application data or legacy application data, respectively).

Here, run-time code comprising network application data is considered run-time code that was generated from a native design-time representation of the application, and run-time code comprising legacy application data is considered run-time code that was generated from a converted design-time representation of the application. The servlet instance or some other related component in Gungabeesoon inherently determines whether the run-time code was generated from a native design-time representation of the application or from a converted design-time representation of the application, so as to respond appropriately. The native and converted design-time representations are further addressed below.

Gungabeesoon does not expressly disclose the run-time code comprising a flag indicating whether the run-time code was generated from a native design-time representation or a converted design-time representation, and does not expressly disclose determining whether the run-time code was generated from the native design-time representation of the application or from the converted design-time representation of the application based on the flag.

Nonetheless, one of ordinary skill in the art could, with predictable results, implement the teachings of Gungabeesoon such that the run-time code comprises such a flag. For example, in an analogous art, Quinn teaches HTML code that includes a flag identifying the application that generated the code (see, for example, column 6, lines 21-62, and column 8, lines 44-63). The flag facilitates the appropriate handling of the code based on the application that generated the code (see, for example, column 5, line 66 to column 6, line 11).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the teachings of Gungabeesoon such that the run-time code comprises a flag indicating whether the run-time code was generated from the native design-time

representation or the converted design-time representation, and such that the determining whether the run-time code was generated from the native design-time representation of the application or from the converted design-time representation of the application is based on the flag, as Quinn suggests.

Gungabeesoon further discloses that:

the native design-time representation of the application is for use in a first run-time environment for executing applications having been developed in a first design-time environment (see, for example, column 9, lines 41-48, which shows a first run-time environment for executing applications developed in a first design-time environment), the first design-time environment using a first programming model comprising one or more first model elements including models, views, and controllers (see, for example, column 8, line 67 to column 9, line 7, which shows that the first design-time environment uses a programming model that comprises views in the form of HTML code and controllers in the form of JavaScript code, and column 9, line 54 to column 10, line 2, which shows that the programming model comprises models in the form of JavaBean data objects); and

the converted design-time representation of the application is generated from an original design-time representation of the application developed for use in a second run-time environment for executing applications having been developed in a second design-time environment (see, for example, column 8, lines 44-54, which shows that the converted design-time representations are generated from original design-time representations of the application in the form of screen definitions, and column 7, lines 50-56, which shows a second run-time environment for executing applications developed in a second design-time environment), the converted design-

time representation is stored as metadata in a metadata repository (see, for example, column 9, lines 41-48 and 54-60, which shows that the converted design-time representation is metadata for application data inserted at run-time, and column 11, lines 2-7, which further shows that the converted design-time representation is stored in a repository for retrieval at run-time), the converted design-time representation having been developed by a development tool based on a metamodel that defines metamodel objects (see, for example, column 8, line 67 to column 9, line 7, and column 9, line 54 to column 19, line 2, which shows that the converted design-time representation is developed based on a metamodel that defines metamodel objects such as HTML code, JavaScript code and JavaBean data objects), the second design-time environment using a second programming model comprising one or more second model elements including screens and processing logic for each screen (see, for example, column 8, lines 12-29, which shows that the second design-time environment uses a programming model comprising screens and processing logic), the original design-time representation including one or more application screens and original processing logic for each application screen (see, for example, column 8, lines 44-47, which shows that the original design-time representation includes one or more application screens), the converted design-time representation including one or more application views based on the one or more application screens, and converted processing logic based on the original processing logic, the converted processing logic capable of being executed in the second run-time environment (see, for example, column 10, lines 31-36, which shows that the converted design-time representation includes one or more application views and processing logic based on the original application screens and processing logic, and column 10, lines 42-49, which shows that the converted processing logic is executable in the second run-time environment);

if the run-time code was generated from the native design-time representation, executing the run-time code in the first run-time environment using a set of run-time modules in the first run-time environment (see, for example, column 10, lines 42-49, which shows executing the run-time code generated from the native design-time representation in the first run-time environment using run-time modules such as the servlet instance in the first run-time environment); and

if the run-time code was generated from the converted design-time representation, executing the run-time code in the first run-time environment using an adapter operable to interface with a set of run-time modules in the second run-time environment (see, for example, column 11, lines 8-22, which shows executing the run-time code generated from the converted design-time representation in the first run-time environment using run-time modules such as the legacy application in the second run-time environment, and column 10, lines 3-17, which shows interfacing with the second run-time environment using an adapter in the form of a Publish-to-Web component in the first run-time environment), comprising using the adapter to perform a function not performed by the original processing logic (see, for example, column 9, lines 17-27, which shows that executing the run-time code comprises using the adapter to perform functions that the original processing logic does not perform).

Gungabeesoon does not expressly disclose using the adapter to perform a function including input validation.

However, Gungabeesoon does teach performing input validation (see, for example, column 8, line 67 to column 9, line 7). One of ordinary skill in the art could, with predictable results, implement the teachings of Gungabeesoon such that the adapter performs the input validation. For example, in an analogous art, Margoscin teaches a middleware program that

functions as an adapter operable to interface with a user interface and a business transaction server (see, for example, the abstract). The middleware program (i.e., the adapter) performs support functions such as data and syntax validation (see, for example, column 3, lines 14-19). Specifically, Margoscin teaches that the middleware program (i.e., the adapter) performs a function comprising input validation (see, for example, column 6, lines 20-36).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the teachings of Gungabeesoon and Quinn such that executing the run-time code comprises using the adapter to perform a function including input validation not performed by the original processing logic, as Margoscin suggests.

With respect to claim 11 (original), the rejection of claim 8 is incorporated, and Gungabeesoon further discloses that:

executing the run-time code using the set of run-time modules in the first run-time environment comprises using a first sequence of process steps (see, for example, column 10, lines 42-49, which shows a first sequence of process steps that comprises, for example, creating a socket and spawning a thread); and

executing the run-time code using the set of run-time modules in the second run-time environment comprises using a second sequence of process steps (see, for example, column 11, lines 8-22, which shows a second sequence of process steps that comprises, for example, forwarding data to the socket).

With respect to claims 12 (currently amended) and 14 (original), the claims are directed to an apparatus that corresponds to the computer program product of claims 1 and 6, respectively (see the rejection of claims 1 and 6 above).

With respect to claims 16 (currently amended) and 18 (original), the claims are directed to a method that corresponds to the computer program product of claims 1 and 6, respectively (see the rejection of claims 1 and 6 above).

6. Claims 2, 9, 13 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gungabeesoon in view of Quinn and Margoscin, as applied to claims 1, 8, 12 and 16 above, respectively, and further in view of "Database Performance in the Real World: TPC-D and SAP R/3" by Doppelhammer et al. (art of record, "Doppelhammer").

With respect to claim 2 (original), the rejection of claim 1 is incorporated. Gungabeesoon discloses that the first programming model is a legacy programming model and the second programming model is a Web-based programming model (see, for example, column 8, lines 5-11), but does not expressly disclose that the first programming model is the SAP Dynpro programming model and the second programming model is the SAP Web Dynpro programming model.

However, Doppelhammer teaches the SAP Dynpro programming model comprising one or more model elements including screens and processing logic for each screen (see, for example, page 125, section 2.3, first paragraph).

Moreover, Gungabeesoon suggests that such programming models are supported (see, for example, column 11, lines 23-27), and further discloses that converting applications from the

legacy programming model to the Web-based programming model enables one to access the applications across the network and take advantage of Web-based technology without having to make code changes to the applications (see, for example, column 11, lines 42-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teachings of Gungabeesoon, Quinn and Margoscin in cases where the first programming model is the SAP Dynpro programming model and the second programming model is the SAP Web Dynpro programming model, so as to take advantage of Web-based technology without having to make code changes to the applications.

With respect to claim 9 (original), the rejection of claim 8 is incorporated. Gungabeesoon discloses that the first programming model is a Web-based programming model and the second programming model is a legacy programming model (see, for example, column 8, lines 5-11), but does not expressly disclose that the first programming model is the SAP Web Dynpro programming model and the second programming model is the SAP Dynpro programming model.

However, Doppelhammer teaches the SAP Dynpro programming model comprising one or more model elements including screens and processing logic for each screen (see, for example, page 125, section 2.3, first paragraph).

Moreover, Gungabeesoon suggests that such programming models are supported (see, for example, column 11, lines 23-27), and further discloses that converting applications from the legacy programming model to the Web-based programming model enables one to access the applications across the network and take advantage of Web-based technology without having to make code changes to the applications (see, for example, column 11, lines 42-55).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to apply the teachings of Gungabeesoon, Quinn and Margoscin in cases where the first programming model is the SAP Dynpro programming model and the second programming model is the SAP Web Dynpro programming model, so as to take advantage of Web-based technology without having to make code changes to the applications.

With respect to claim 13 (original), the claim is directed to an apparatus that corresponds to the computer program product of claim 2 (see the rejection of claim 2 above).

With respect to claim 17 (original), the claim is directed to a method that corresponds to the computer program product of claim 2 (see the rejection of claim 2 above).

Conclusion

7. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is 571-272-3707. The examiner can normally be reached on Monday to Friday from 8:00 AM to 4:30 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192

Michael J. Yigdall
Examiner
Art Unit 2192

/MY/